AMENDMENTS TO THE SPECIFICATION:

Please amend the paragraph beginning on page 4, line 4 as follows:

--Considering the foregoing, it is highly desirable to provide a flowcharting technique for simplifying a view of any complex data that has linked data elements. It is further very desirable to provide a flowcharting technique for processing processes or workflows that are too large and complex for conventional flowcharting techniques and are constantly changing. Still further, it is highly desirable to provide a flowcharting technique that presents a sequence of processing operations clearly. Yet further it is very desirable to provide an automated flowcharting technique for processing processes or workflows, thereby eliminating a need for an employee dedicated to updating flowcharts of the processes or workflows.--

Please amend the paragraph from page 4, line 14 – page 5, line 2 as follows:

-- As aforementioned, if a single flowchart were to be drawn depicting a complex process or workflow, it's tremendous size--both visually and physically--would make the flowchart totally unmanageable, particularly for presentation over an Intranet/Internet where bandwidth is at a premium. Because of bandwidth considerations for the Intranet/Internet, it is highly desirable to provide a flowcharting technique for processing processes or workflows for presentation over a communications network, such as the Intranet/Internet. In light of this consideration, it is highly desirable to provide a flowcharting technique for processing processes or workflows for presentation over a communications network via a web-enabled browser, such as Netscape Communicator™ or Internet Explorer™. That is, it is highly desirable to provide a flowcharting technique for processing processes or workflows, which is capable of generating markup language flowcharts (e.g., HTML) for transmission over the communications network. --

Please amend the paragraph at page 8, beginning at line 1 as follows:

-- According to yet another aspect of the present invention, there is provided a program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform an automatic flowcharting method for diagrammatically representing a multi-nodal process comprising processing operations and decision operations, the method comprising: converting processing operations and decision operations of the multi-modal process into a data structure; analyzing the data structure for identifying a first group of processing operations that appear once in the data structure, and for identifying a second group of processing operations that are associated with two or more decision operations in the data structure; traversing the data structure to generate an ordered sequence of processing operations for visual representation; and generating a diagrammatic representation of the ordered sequence including orienting the processing operations in a vertical dimension and associating attributes to each processing operation of the processing operations according to their identified group while offsetting each successive processing operation of the processing operations in a horizontal dimension, and linking each processing operation of the second group to a further processing operation of the processing operations according to a decision operation of the two or more decision operations.--

Please amend the paragraph beginning at page 14, line 20 as follows:

--The subsidiary function Get_Places_To_Go( ) determines a particular number of processing operations to which the processing operation in the from_op variable can go and not go (e.g., branching). That is, this function determines how many processing operations in op_struct.next_ppo_id described above, to which the current processing operation in the from_op

branches. This is done by analyzing in a loop whether op_struct.ppo_id for every element in the original data structure is equal to the from_op variable. If op_struct.ppo_id = from_op, the program stores the processing operations to which the processing operation in the from_op variable ~~braches~~ branches to, in a one-dimensional array called "nl_op" (i.e., next level operations). To access elements in the nl_op array, the program defines a counter called nlc (i.e., next level counter), which is initialized to 1. The foregoing value of from_op utilized in first iteration of the Get_Places_To_Go( ) function is "01_BRD_INGATE" (See Figure 2, record 209, PPO_ID 210). Consequently, the next level operation is nl_op[nlc] = op_struct.next_ppo_id (i.e., "02_BENCH_MET_TST" for nlc=1). Furthermore, within this function, a check is performed to determine whether next_ppo_id (i.e., "02_BENCH_MET_TST") is in the vplace array described hereinabove. Because "02_BENCH_MET_TST" is not in the vplace array, the counter total_place is incremented by one (i.e., total_place=2), wherein vplace[total_place] = "02_BENCH_MET_TST". Furthermore, a new Boolean array called "followPath" is created for each from_op, to store results of the search through the visited place (i.e., processing operations) in the vplace array. Since "02_BENCH_MET_TST" has not yet been analyzed at this point, followPath[nlc]=true. If the "02_BENCH_MET_TST" processing operation were already in the vplace array the value of followPath[nlc]=false.--

Please amend the paragraph beginning at page 18, line 1 as follows:

--Based on a processing operation that is to be followed (i.e., followPath[2] = "03_STENCIL_BOT") as indicated in the followPath array, the program reorders the vfile array (i.e., elements 5 and 6 indicated above) to according to the processing operation that should be followed, which is stored after the one not to be followed (i.e., "05_BOTTOMSIDE") . It should be noted that the reordering according to followPath array may also be conveniently performed

after all processing operations have been written to the vfile array. Therefore, based on the FollowPath array, the program reorders the two elements in the following manner:

vfile[5]= ". . . . 04_PASTE_VISUAL"
vfile[6]= ". . . . 03_STENCIL_BOT"

--

Please amend the paragraph beginning at page 19, line 18 as follows:

-- Further referring to Figure 3, a preferred way for determining the horizontal indentation (e.g., offset) and vertical alignment of the successive processing operations is by using a logical dot-count-by-row construct 410, which is illustrated in Figures 4 and 5. In this construct, as illustrated in Figure 4, each successive processing operation 410...420 (i.e., successive row) is logically horizontally indented or offset a length of one additional dot for each successive row. Yet further, processing operations to which a link, such as link 307, is defined get the dot count equal to the originating processing operation, such as operation 414. The length for the dot is predetermined, and can be set [[by]] to a particular number of characters in length. For example, the length of one dot can be set to two, three or four characters in length, depending on particular horizontal compactness of the desired diagrammatic representation. --